Das Projekt IO-Broker macht Smart-Home-Anwendungen kompatibel

Vermittlerdienste

In der stetig wachsenden Smart-Home-Welt beziehungsweise im Internet der Dinge vermittelt IO-Broker zwischen Einzelsystemen wie der Broker in der Finanzwelt zwischen Kunde und Bank. Stefan Heinle

© Luca Bertolli, 123RF

IO-Broker [1] ist aus CCU.IO entstanden, einem erfolgreichen Projekt der deutschen Homematic-Community, das diese aber nicht mehr weiterentwickelt. Von Grund auf neu programmiert ist IO-Broker deutlich leistungsfähiger und universeller als sein Vorgänger. Erstmalig vorgestellt im August 2014 erfreut es sich zunehmender Beliebtheit. Das hat gute Gründe, denn inzwischen sind bereits im Basis-Setup mehr als 100 Interfaces zu Smart-Home-Systemen, intelligenten Haushaltsgeräten sowie Produkten der Unterhaltungselektronik und diversen Webservices enthalten.

Die Adapter

Im IO-Broker-Jargon heißen diese Interfaces Adapter. Eine Auswahl der interessantesten Adapter ist in Tabelle 1 zusammengestellt. Mit Kenntnissen in Javascript oder Node.js darf jeder Anwender auch eigene Adapter entwickeln [2]. Aber selbst ohne Programmierkenntnisse stehen die Chancen recht gut, die eigene heterogene Smart-Home-Infrastruktur allein über bereits existierende Adapter in IO-Broker abzubilden.

Die Installation sowie die Aktualisierung von Adaptern bewerkstelligt IO-Broker komfortabel über seine Web-Administrationsoberfläche selbst (Abbildung 1). Im Hintergrund erledigt der Javascript Package Manager NPM [3] die eigentliche Arbeit und bindet automatisch die notwendigen Pakete und deren Abhängigkeiten mit ein.

Die meisten Adapter bringen ein kleines Konfigurations-Dialogfenster mit, in dem der Anwender generelle Einstellungen zum Interface vornimmt. Abbildung 2 zeigt als Beispiel die möglichen Parameter des Smartmeter-Adapters. Adapter dürfen mehrfach instanziert werden, was für obiges Beispiel auch die Abfrage mehrerer Stromzähler möglich macht. Jede Instanz ist separat konfigurierbar und besitzt eigene Zustände und Objekte innerhalb von IO-Broker.

Systemanforderungen

Wie es in der Smart-Home-Welt üblich ist, benötigt der Betrieb von IO-Broker kein schweres Gerät. Bereits ein Raspberry PI (Modell 2) mit 1 GByte RAM reicht für eine ordentliche Performance aus, x86-Plattformen mit Windows, Linux oder auch Mac OS X funktionieren ebenso. Einzige Voraussetzung ist, dass das Zielbetriebssystem Node.js [4] (die Server-seitige Javascript-Laufzeitumgebung) unterstützt.

Arbeitsspeicher von weniger als 1 GByte empfiehlt sich nicht, da IO-Broker extensiven Gebrauch von In-Memory-Speicherung von Json-Files macht, um Objekte, Zustände und Ereignisse abzubilden. Eine optionale Optimierung wäre der Einsatz der In-Memory-Datenbank Redis [5]. Zudem startet für jede Adapterinstanz ein neuer Node.js-Prozess. Der Speicherbedarf eines einzelnen Adapters liegt damit im Bereich von 10 bis 60 MBytes.

Der Autor

Stefan Heinle ist Diplom-Ingenieur der Elektrotechnik und begeisterter Heimautomatisierer. Er ist zertifizierter KNX-Partner, Autor des Buches „Heimautomation mit KNX, DALI, 1-Wire und Co." (2015, Rheinwerk-Verlag, ISBN 978-3-8362-3461-0) und bietet auf der Seite [www.heimautomation-buch.de] regelmäßig Zusatzinformationen zum Buch sowie einen professionellen Smart-Home-Planungsservice.

Eine Besonderheit stellt der so genannte Multihost-Modus dar. Damit ist gemeint, dass IO-Broker in der Lage ist, die Rechenlast auf mehrere Server zu verteilen. Sollte also irgendwann der ursprüngliche Rechner zu langsam sein, lässt sich einfach ein weiterer Host mit IO-Broker nachrüsten. Über die Weboberfläche ordnet der Administrator dann die Adapter einem der Server im Verbund zu. So entsteht eine skalierbare IT-Infrastruktur für das eigene Smart Home.

Installation und Start

Der einfachste Weg zum eigenen IO-Broker-Server führt über vorgefertigte Images. Die Projekt-Webseite bietet im Download-Bereich zahlreiche Images für die gängigen Kleinstrechner Raspberry Pi 1, 2, 3, Banana Pi, Pine64, Cubieboard 3, Cubox und Linux-Distributionen an. Daneben gibt es eine Windows-Version als Setup-Exe-File. Eine Nachinstallation in ein laufendes Linux-System ist auch möglich, ebenso der Betrieb in einer virtuellen Maschine oder in einem Docker-Container. Das IO-Broker-Team stellt dazu auf der Projektwebseite [1] einige Setup-Anleitungen bereit.

Es lohnt sich, beim ersten Start von IO-Broker das Angebot des Einrichtungsassistenten anzunehmen, der das lokale Netzwerk und lokale Schnittstellen nach existierenden Geräten scannt und passende Adapter vorschlägt. Direkt nach der Installation ist IO-Broker bereits lauffähig und präsentiert über den eingebauten Webserver auf Port 8081 die Administrationsoberfläche.

Der Reiter »Adapter« zeigt alle einbindbaren Interfaces mit einer Kurzbeschreibung und Versionsnummer. Der Fragezeichen-Knopf in der Spalte »Installieren« führt zu einer ausführlicheren Beschreibung des Adapters, der »Plus«-Button installiert ihn oder fügt eine weitere Instanz hinzu. Im Reiter »Instanzen« sind alle installierten Adapter-Instanzen aufgeführt (Abbildung 3). Sie lassen sich von dort aus konfigurieren, starten, stoppen und auch wieder entfernen. Ein Ampel-Symbol am linken Rand gibt Auskunft über den Zustand einer Instanz.

Über die weiteren Spalten kann der Anwender die aktuelle Log-Stufe verändern sowie die Anzahl der generierten

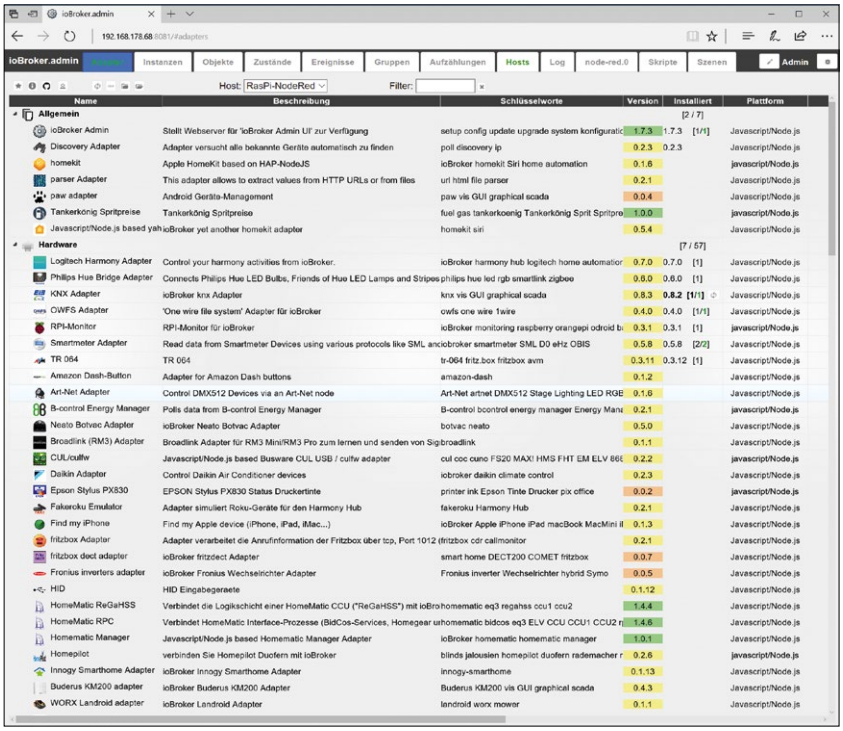

Abbildung 1: Die Weboberfläche für die Administration von IO-Broker.

Ereignisse und deren aktuellen RAM-Verbrauch pro Adapter einsehen. Dazu müssen aber die Experten-Einstellungen aktiviert sein (kleines Icon links oben).

Objekte, Zustände und Ereignisse

Ist ein Adapter konfiguriert und gestartet, erzeugt er automatisch eine Reihe verschiedener Objekte, die eigentlichen Datenspeicher von IO-Broker. Alle angelegten Objekte werden im gleichnamigen Reiter angezeigt und automatisch aktualisiert. Mit welcher Frequenz die Aktualisierung erfolgt, bestimmt entweder die Adapter-Konfiguration oder es ergibt sich aus der Update-Rate des Senders.

Die Objekt-Ansicht zeigt eine hierarchische Struktur, die die Orientierung

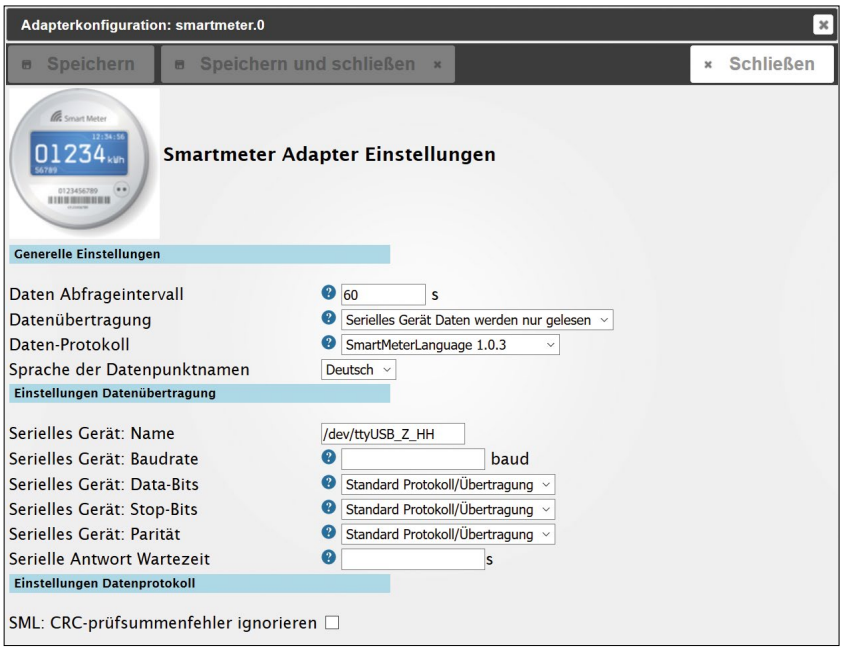

Abbildung 2: Adapterkonfiguration am Beispiel des Smartmeter-Adapters.

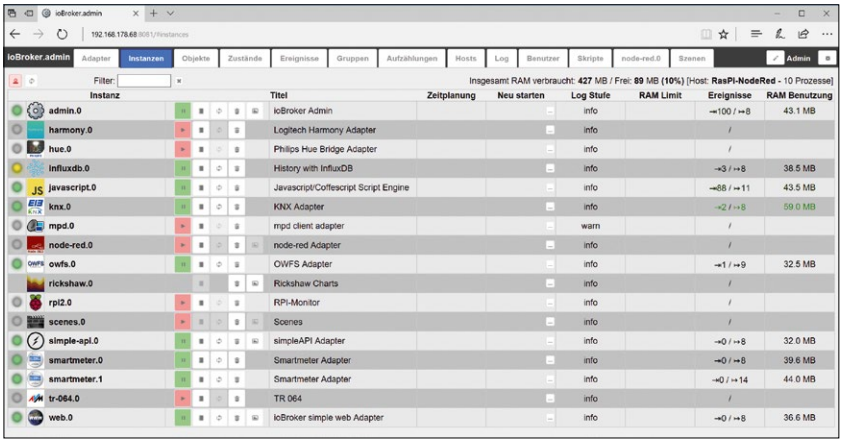

Abbildung 3: Adapter-Instanzen mit Status-Informationen.

deutlich erleichtert. In der ersten Hierarchiestufe untergliedert IO-Broker die Objekte nach der Instanz des zugrunde liegenden Adapters. Darunter entscheidet sich die Baumstruktur je nach Adapter. So existiert für eine Influx-DB-Instanz beispielsweise nur ein einziges Objekt, das den aktuellen Verbindungsstatus zur Datenbank anzeigt.

Die Objektstruktur des Adapters für den Music Player Daemon (MPD) enthält dagegen über 100 Objekte, die unter anderem umfangreiche Informationen über den gerade abgespielten Musiktitel bereitstellen, die aktuelle Playlist auslesen oder sogar den MPD-Server fernsteuern können.

Die Integration des in der Gebäude-Automation wichtigsten Bussystems KNX [6] gelingt ebenso komfortabel über den passenden, gleichnamigen Adapter und dessen Objekte. Die Objekte legt der Adapter automatisch an, und zwar auf Basis einer eingelesenen ».knxproj«-Datei, die wiederum aus der KNX-Integrationssoftware ETS entsteht. Um mit dem KNX-Bus zu kommunizieren, benötigt der Adapter nur noch die IP-Adresse des KNX-IP-Gateway, die Port-Nummer 3671 und die physikalische KNX-Adresse des Gateway. Für jede Gruppenadresse [7], die aus dem ETS-Exportfile ausgelesen wird, enthält der KNX-Objektbaum in IO-Broker – nach der Adapter-Initialisierung – ein einzelnes Objekt (Abbildung 4).

Der KNX-Adapter versucht durch eine Art Ähnlichkeitsprüfung aller Gruppen-Adressnamen herauszufinden, welche Schaltadressen und Statusrückmeldungen zusammengehören. Eine sauber gepflegte ETS-Datenbank ist zwar kein Erfolgsgarant, erleichtert dem Adapter aber die Arbeit. Aktuell unterstützt IO-Broker noch nicht alle Datenpunkttypen (DPT) des KNX-Standards und nicht jeder ETS-Export wird korrekt in IO-Broker eingelesen, aber der Grundstein für eine nahtlose Integration ist bereits gelegt.

Tabelle 1: Auswahl an Adaptern

Adapter	Funktion
Discovery	Findet Geräte im lokalen Netzwerk und schlägt Adapter vor
FHEM	Verbreitete Automatisierungsplattform
Homematic	Anbindung an das weitverbreitete Homematic-System
iCal	Liest Kalender-Files im ».ics« Format
Influx-DB	Logging von Zuständen in einer Influx DB
Javascript/Node.js	Eigene Javascript-Skripte ausführen
KNX	Zugriff auf KNX-Bus
Modbus	Modbus-Verbindung, Slave oder Master
MPD	Ansteuerung des Music Player Daemon
MQTT	Kommunikation mit MQTT-Broker
Mysensors	Hw/Sw-DIY Projekt für günstige IoT-Sensorik/Aktorik
Netatmo	Zugriff auf Netatmo-Smarthome-Geräte
Open HAB	Brandneues Interface für Open HAB
Osram Lightify	Smarte Beleuchtungsprodukte, ähnlich Philips Hue
OWFS	Zugriff auf 1-Wire-Bus, über One-Wire-Filesystem
Philips Hue Bridge	Kommunikation mit der Philips-Hue-Bridge
Pushbullet	Messenger-App
RWE Smarthome	Einbindung von RWE-Smarthome-Geräten
Scenes	Szenen innerhalb IO-Broker erstellen und aktivieren
Smartmeter	Abfrage von Stromzählern über SML-Protokoll oder DO
Sonos	Abfragen und steuern von Sonos-Playern
Telegram	Kommunikation mit Telegram-App
TR 064	Steuerung/Abfrage von AVM-Fritzboxen
VIS	Visualisierungstool für eigene Bedienoberflächen
ZWave	Anbindung an Z-Wave-Funknetz (Openzwave-Paket)

Datenlogging mit Influx DB

Das komfortable Datenlogging mit dem Influx-DB-Adapter wäre allein schon Grund genug, IO-Broker im eigenen Smart Home einzusetzen. Bei Influx DB [8] handelt es sich um eine Open-Source-Datenbank – spezialisiert auf das Speichern, Analysieren und Darstellen von zeitbasierten Messwerten, wie geschaffen also für die Ablage von IO-Broker-Objekten. Einzige Voraussetzung ist ein aufgesetzter Influx-DB-Server, der natürlich auf demselben Rechner laufen darf wie IO-Broker selbst.

Bei der Adapter-Konfiguration werden die Zugangsdaten zum Datenbankserver, die gewünschte Datenbank für die Ablage der Messwerte sowie einige Details zur Datenspeicherung selbst vorgegeben. Fortan erscheint zusätzlich bei allen Objekten im Reiter »Objekte« ein kleines Zahnrad-Symbol über das ein Konfigurations-Fenster für die Optionen der Datenspeicherung zum gewählten Objekt geöffnet wird (Abbildung 5).

Durch Abhaken der »Aktiviert«-Checkbox im Reiter »Einstellungen« startet die Aufzeichnung – schon wandern künftig die gewonnenen Messwerte zusätzlich

in eine Influx-Datenbank. Im Reiter »Tabelle« finden sich zurückgelesene letzte Werte zusammen mit deren Zeitstempel. Soll es nicht nur eine textuelle Ausgabe sein, sondern eine Grafik, hilft der dritte Reiter »Grafik« weiter. Er nimmt seine Arbeit jedoch erst dann auf, wenn der Admin vorher den konfigurationslosen Adapter »Flot« [9] oder »Rickshaw Charts« (veraltet, [10]) installiert hat. Dann nämlich generiert IO-Broker auch gleich noch den passenden xy-Plot zu einer gespeicherten Messreihe.

Abbildung 6 zeigt den automatisch erzeugten Plot des Stromverbrauchs im Zeitverlauf, erfasst über den Smartmeter-Adapter, weitergegeben via KNX-Adapter, protokolliert durch den Influx-DB-Adapter und letztendlich dargestellt mit Hilfe des Rickshaw-Adapters. Genau so sollte eine modulare Smart-Home-Plattform aufgebaut sein.

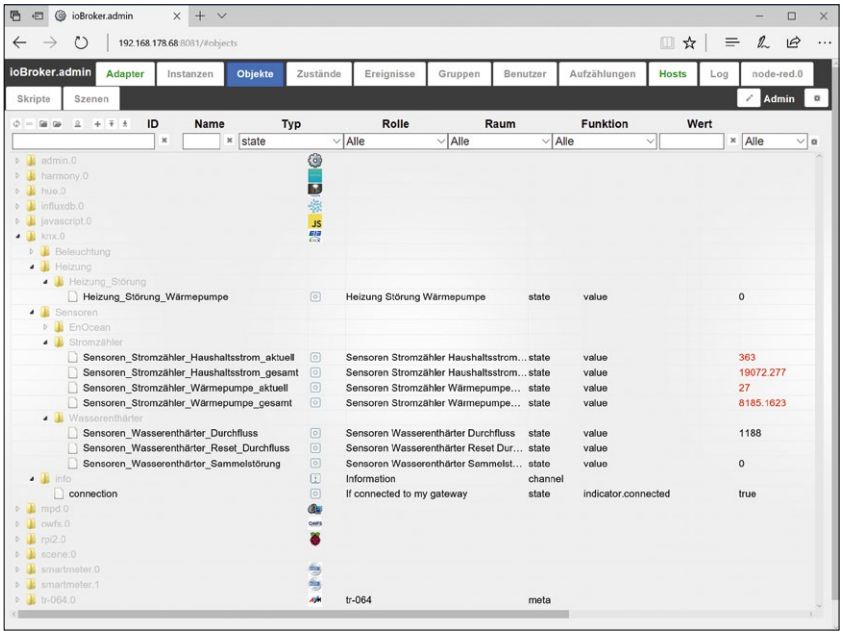

Abbildung 4: IO-Broker-Objekte, automatisch aus KNX-Gruppenadressen erzeugt.

Automatisierung mit IO-Broker

Das Einlesen von Daten und Messwerten über die instanzierten Adapter macht allein noch keine Automatisierung aus. Dazu muss das System die gesammelten Daten zusätzlich auswerten, verknüpfen, verarbeiten und anschließend über andere Adapter wieder ausgeben. IO-Broker bietet dazu verschiedene Mechanismen an, die auch in Kombination verwendet werden können. Diese sind, in aufsteigender Komplexität:

- Das Erstellen von Szenen mit dem Adapter »Scenes«.
- Die grafische Programmierung über die Einbindung des Node-Red-Adapters [11].
- Eigene Funktionen in Javascript mit dem Adapter »Javascript/Coffeescript Script Engine«.

Möglichkeit 1: Szenen

Die einfachste Form der Automatisierung sind die IO-Broker-Szenen. Ist der Szenen-Adapter »Scenes« installiert und aktiviert, blendet IO-Broker über das Webinterface einen neuen Reiter namens »Szenen« ein. Das Anlegen einer Szene (Abbildung 7) geschieht über zwei Schritte: das Anlegen und Konfigurieren der Szene selbst und das Hinzufügen von beteiligten Datenpunkten. Das Plus-Symbol legt eine leere Szene an, über das Buch-Symbol wird sie anschließend konfiguriert.

Eine Szene benötigt immer eine Trigger-Bedingung, die darüber entscheidet, wann die Szene ausgeführt werden soll. Das Beispiel in Abbildung 8 etwa würde immer dann auslösen, wenn ein eingehender Anruf erkannt wird. Zu diesem Zweck greift IO-Broker über das TR064-Protokoll auf den Call-Monitor einer Fritzbox im Heimnetz zu, dessen Zustand »ringing« den Wert »true« annimmt, wenn das Telefon klingelt.

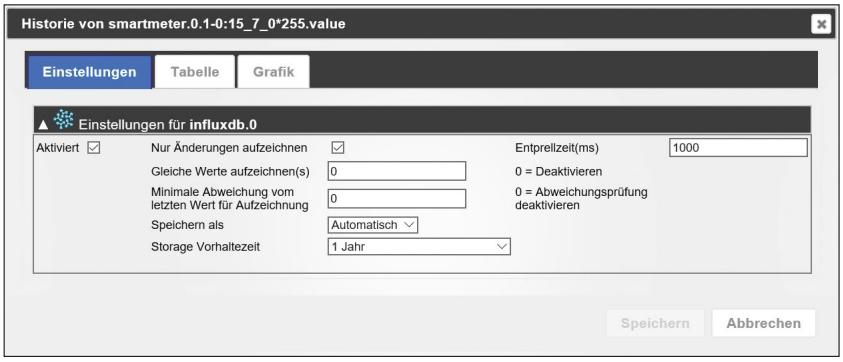

Abbildung 5: Datenlogging leicht gemacht – der Influx-DB-History-Adapter.

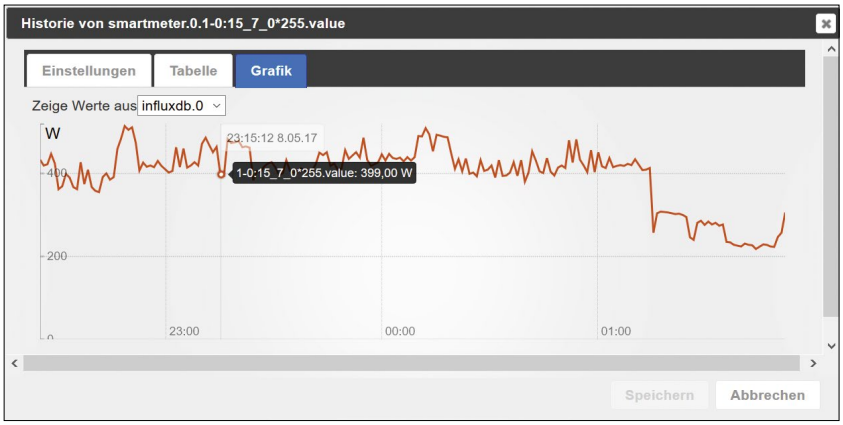

Abbildung 6: Der Rickshaw-Adapter erstellt einfache Plots von IO-Broker-Objekten.

Abbildung 7: Szene für Stummschaltung des Webradios bei eingehendem Anruf.

Die im zweiten Schritt hinzuzufügenden Datenpunkte übernehmen das Schalten von Aktionen, sobald die Szene ausgelöst wird. Das Plus-Symbol am rechten Rand der Szene hilft dabei, einen oder mehrere Datenpunkte der Szene zuzuordnen. Über die ID des Datenpunkts legt der Anwender das Objekt fest, das zu schalten ist. Für den obigen Anwendungsfall ist beispielsweise die ID »mpd.0.pause« geeignet, die einem konfigurierten MPD-Server das Kommando zum Pausieren der aktuellen Audio-Ausgabe gibt.

Möglichkeit 2: Flows in Node-Red

Eine weitere Möglichkeit, die IO-Broker-Objekte in eigenen Logik-Funktionen zu verwenden, bietet die eingebaute Schnittstelle zu Node-Red [11]. Mit Node-Red, einem Browser-basierten, grafischen Editor für IoT-Anwendungen, lassen sich recht einfach Datenflüsse modellieren und Funktionen ausführen. Voraussetzung für die Kommunikation mit Node-Red ist die Installation und Aktivierung des Node-Red-Adapters.

IO-Broker erzeugt für die Node-Red-Flows einen eigenen Reiter und fügt der Node-Bibliothek gleich noch ein paar zusätzliche Nodes zum Datenaustausch hinzu. Innerhalb der Node-Red-Flows dienen diese Input- und Output-Nodes IO-Broker dem Lesen beziehungsweise Schreiben von IO-Broker-Objekten.

In Kombination mit dem ständig wachsenden Pool an nützlichen Nodes lassen sich damit bereits recht anspruchsvolle Logik-Aufgaben lösen. Node.js-Kenntnisse sind dabei von Vorteil, aber nicht unbedingt Voraussetzung.

Abbildung 9 zeigt einen einfachen Flow in Node-Red zusammen mit den IO-Broker-Nodes für den Datenaustausch zwischen beiden Systemen. Im Beispiel reagiert der Flow auf einen Anstieg der Rack-Temperatur über 45 Grad und visualisiert dieses Ereignis durch Anschalten einer Hue-Beleuchtung. Die eigentliche Kommunikation mit der Hardware übernimmt dabei IO-Broker.

Die Rack-Temperatur misst in diesem Fall ein 1-Wire-Sensor und bildet sie über den OWFS-Adapter als Objekt in IO-Broker ab. Die Kommunikation mit der Hue-Bridge und damit auch die Ansteuerung einer Leuchte gelingt über den Adapter »Philips Hue Bridge«.

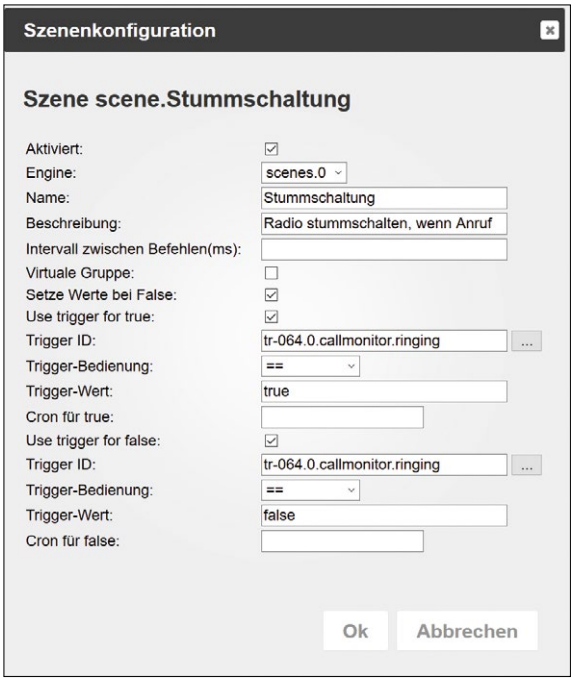

Abbildung 8: Definition einer Szene für Anruferkennung.

Möglichkeit 3: Javascript

Anwender mit soliden Grundkenntnissen in Javascript werden vermutlich die flexibelste Methode der Automatisierung innerhalb IO-Broker nutzen, den Adapter »Javascript/Coffeescript Script Engine«. Er lässt sich auch mehrfach instanzieren, was den Vorteil hat, mehr als eine Laufzeitumgebung für eigene Skripte zu schaffen.

So könnte die erste Instanz »javascript.0« die Produktiv-Skripte beinhalten und »javascript.1« als zweite Instanz für Skripte im Testbetrieb eingesetzt werden. Das Stoppen der Test-Instanz hätte somit keine Auswirkung auf die Produktiv-Instanz. Innerhalb einer Instanz dürfen beliebig viele Skripte angelegt und aktiviert werden. Der eingebaute Editor erleichtert die Code-Eingabe durch Syntax Highlighting und Syntax Checking.

Zudem stehen dem Skriptprogrammierer aber rund 40 IO-Broker-spezifische Funktionen zur Verfügung, um IO-Broker-Objekte zu lesen und zu schreiben, neue Zustände zu erzeugen, diese zu setzen oder abzufragen, andere Skripte zu starten, historische Werte zu ermitteln, Daten zu formatieren und vieles mehr. Was die zeitliche Steuerung betrifft, hält die Scripting Engine von IO-Broker mehrere Möglichkeiten offen:

- Callback-Funktionen, die IO-Broker mit Eintreten eines Ereignisses (zum Beispiel der Änderung eines Zustandes) ausführt.
- Angabe des Ausführungszeitpunktes und der Wiederholungen in Cron-Syntax.

Das Skript in Abbildung 10 verwendet die Funktion »on()«, um bei Zustands-

änderungen der Smartmeter-Messergebnisse (Parameter 1) den Messwert selbst über eine KNX-Gruppenadresse beziehungsweise dessen Objekt-Abbildung in IO-Broker (Parameter 2) in ein KNX-System zu übertragen. Diese Verknüpfung wird insgesamt viermal für die vier unterschiedlichen Messwerte angelegt.

Grafiken

Auch an die Freunde von visuellen Programmierhilfen haben die IO-Broker-Entwickler gedacht und der Scripting Engine eine Blockly-Unterstützung spendiert. Bei Blockly [12] handelt es sich um eine Bibliothek, die visuelle Programmier-Editoren in eigene Applikationen einbinden hilft. Blockly nutzt visuelle Codeblöcke, die der Entwickler sozusagen ineinanderstecken. Die Blöcke besitzen Anschlüsse für die Ein- und Ausgabe und sind farblich so strukturiert, dass der Einsatzzweck mit ein bisschen Übung abgelesen werden kann. Schön ist: Blockly erzeugt stets syntaktisch korrekten Code, auch ohne dass der Anwender mit der Javascript-Syntax vertraut sein muss.

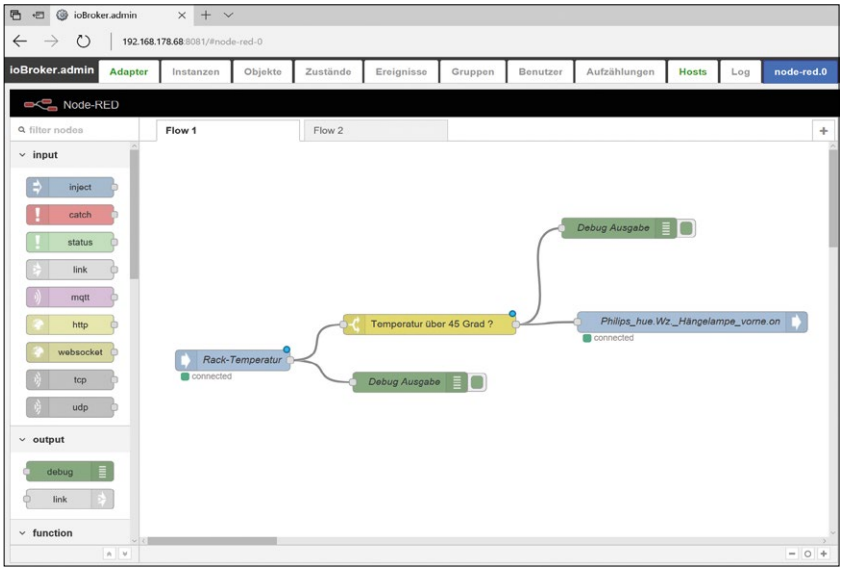

Abbildung 9: Umsetzung einer kleinen Logik in Node-Red.

IO-Broker ist nicht nur ein mächtiges Werkzeug, um unterschiedliche Smart-Home-Systeme zu verbinden und mit eigenen Automatisierungsregeln zu er-

Anzeige

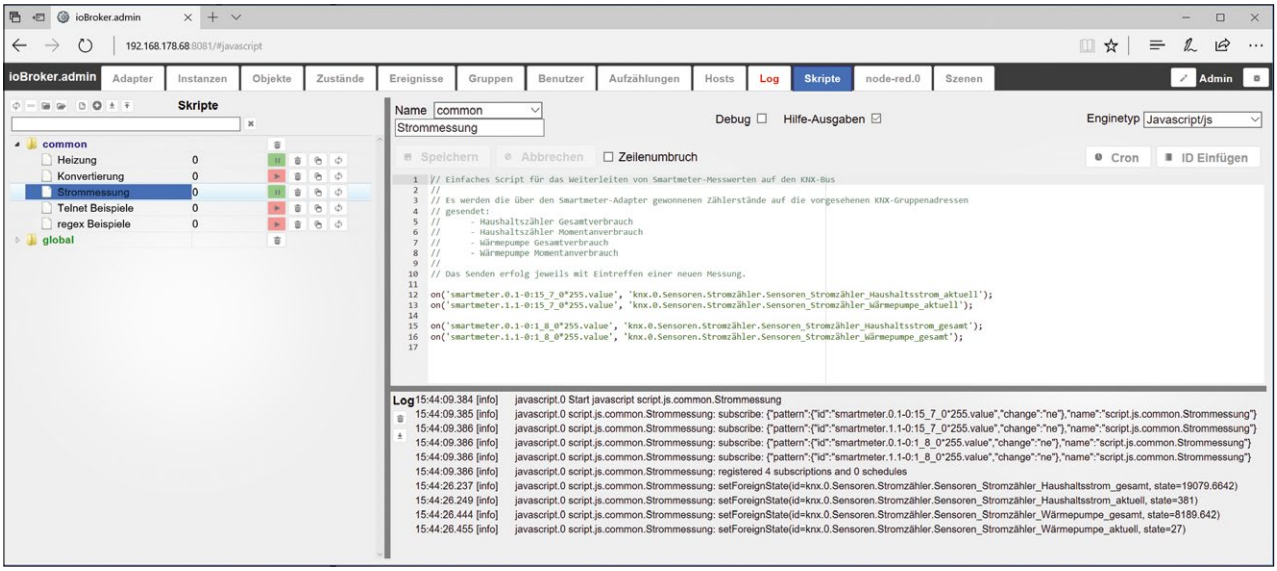

Abbildung 10: Javascript zur Weitergabe von Smartmeter-Werten auf KNX-Bus-Gruppenadressen.

weitern, sondern es bietet auch einen leistungsfähigen Editor für Visualisierungsoberflächen [13].

Der Editor ist betriebsbereit, sobald die optionalen Adapter »IO-Broker Simple Web Adapter« und »IO-Broker Visualisation« installiert sind, und lässt sich über die URL »*IP_des_IO-Broker_Servers*:8082/vis/« aufrufen. Beim ersten Start begrüßt eine Demo-Visualisierung den Anwender (Abbildung 11).

Jedes Visualisierungs-Projekt kann mehrere unterschiedliche Ansichten (Views) besitzen, die sich wiederum aus einzelnen, frei platzierbaren Widgets zur Anzeige oder Bedienung zusammensetzen. Ein Widget besitzt in der Regel einen Bezug zu einer IO-Broker-Objekt-ID und kann den Objektstatus darstellen oder verändern. Ein einfacher Button ist dabei nur eine von vielen Möglichkeiten. Die Palette reicht von Schiebereglern über Radio-Buttons bis zu Wertelisten und sich verändernden Icons.

Bei Bedarf lassen sich jederzeit weitere Widgets importieren oder über die Webadmin-Oberfläche von IO-Broker nachinstallieren. Der Schritt zur eigenen Visualisierung im Metro-Style ist ebenso klein wie der zum futuristischen Star-Trek-Design. Echte Fans greifen daher zielsicher zum »Icars style Widgets«-Paket.

Fazit

Die Smart-Home-Welt ist äußerst vielschichtig, ihre vielen Systeme und Protokolle sind in der Regel zueinander nicht kompatibel. An dieser Stelle leistet der plattformübergreifende IO-Broker mit seinem erweiterbaren Adapter-Konzept und dem Auto-Discovery erstaunliche Dienste. Er kombiniert unterschiedliche Systeme zu einem großen Ganzen, sorgt für eine gemeinsame Datenbasis, löst komplexe Logik-Aufgaben und bietet ganz nebenbei noch eine leistungsstarke Visualisierung.

Sicherlich bekommt das interessierte Publikum in nächster Zeit noch viele wertvolle Erweiterungen zu sehen – der sympathischen und aktiven IO-Broker-Community sei Dank. *(jcb)* ■

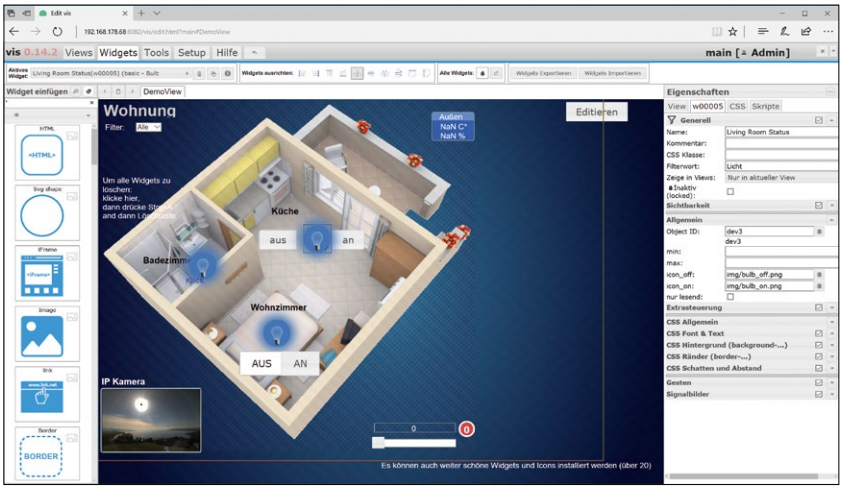

Abbildung 11: Der mächtige Visualisierungseditor von IO-Broker.

Infos

[1] Projekt-Homepage: [http://www.iobroker.net]
[2] Adapter entwickeln: [https://github.com/ioBroker/ioBroker/wiki/Adapter-Development-Documentation]
[3] NPM: [https://www.npmjs.com]
[4] Node.js: [https://nodejs.org]
[5] Redis: [https://redis.io]
[6] KNX Association: [https://www.knx.org]
[7] Stefan Heinle, „KNX – Bussystem für intelligente Gebäudetechnik“: Linux-Magazin 12/16, S. 30
[8] Influx DB: [https://www.influxdata.com]
[9] Flot: [http://www.flogtcharts.org]
[10] Rickshaw: [http://code.shutterstock.com/rickshaw]
[11] Node-Red: [https://nodered.org]
[12] Blockly: [https://developers.google.com/blockly]
[13] IO-Broker.vis-Beispiele: [https://iobroker.net:8080]